Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently Amended) Method for controlling program execution integrity by verifying execution traces, ~~characterized in that it comprises~~ comprising:

- updating a trace print representing an execution pathway and/or handled data on program execution,
- comparing said trace print (current value, calculated dynamically) with an expected value (fixed statically, equal to ~~the~~ a value the trace print should have if program execution is not disturbed) at determined points of the program,
- performing special treatment if the current trace print differs from the expected value.

2. (Currently Amended) Method as in claim 1, ~~characterized in that~~ wherein the special treatment of the program if the current trace print differs from the expected value, consists of securitizing certain data and/or alerting a user of the ill-functioning by a sound or visual signal and/or

interrupting the execution of said program whether definitively or not.

3. (Currently Amended) Method as in claim 1, ~~characterized in that~~ wherein said trace print only concerns critical code fragments of the program and/or program status which is considered critical.

4. (Currently Amended) Method as in claim 1, ~~characterized in that~~ wherein said trace print is calculated incrementally along the execution pathway of the program by successive composition of a function of which one argument is the current trace print value and another argument is a specific observation data item at ~~the~~ point and time of trace print updating (program status and/or program execution point and/or handled data).

5. (Currently Amended) Method as in claim 4, ~~characterized in that~~ wherein said function consists of one of the following functions: « checksum », linear congruency, cyclic redundancy check (CRC), cryptographic tracing print (« digest »), or combination of the following operations: addition, subtraction, «or» exclusive logic (« xor ») with a

constant or with said observation data item; rotation of a constant number of bits; multiplication by an uneven constant.

6. (Currently Amended) Method as in claim 1, ~~characterized in that~~ wherein the trace print is adjusted along the execution pathways before reaching certain points of convergence of ~~the~~ a check flow so that ~~the~~ trace prints of converging pathways are made equal.

7. (Currently Amended) Method as in claim 6, ~~characterized in that~~ wherein the adjustment operation consists of a combination of the following functions: assignment to a constant value, addition with a constant, «or» exclusive logic (« xor») with a constant value.

8. (Currently Amended) Method as in claim 1, ~~characterized in that~~ wherein, at certain points of the program, the trace print is assigned to a certain value rather than deducted from ~~the~~ a preceding trace print value.

9. (Currently Amended) Method as in claim 8, ~~characterized in that~~ wherein said program points are those where execution branches converge whose number is greater than a certain threshold and/or those which are ~~the~~ entry points of

subroutines and/or of exception handlers, and in that said assigned value is a given value and/or any value determined by random drawing and/or a program expression determined by previous analysis as an invariant at the program point under consideration.

10. (Currently Amended) Method as in claim 1, ~~characterized in that~~ wherein the trace print value is compared with the expected value at program points determined by their particular characteristic in ~~the~~ a check flow graph of said program and/or by the type of operations performed at said program points.

11. (Currently Amended) Method as in claim 10, ~~characterized in that~~ wherein said program points are located after each branch and/or before each join of the check flow and/or before each operation which writes in non-volatile memory and/or before certain cryptographic operations and/or before ~~the~~ a call to certain library routines and/or after ~~the~~ a call to certain library routines.

12. (Currently Amended) Method as claimed in ~~any of claims 1 to 11~~ claim 1, ~~characterized in that~~ wherein trace

print setting (calculation and/or updating and/or adjustment and/or assignment) and/or trace print controlling are made:

- explicitly by an instrumentation of the program ~~code, and/or~~
- explicitly by the execution machine (virtual machine and/or processor of the execution platform), on the basis of complementary program data which indicate to said execution machine at which program points and/or with which values (including values resulting from complex operations) the trace print setting and/or controlling operations are to be made, and/or
- implicitly by the execution machine (virtual machine and/or processor of the execution platform), on the basis of a particular observation of executed instructions.

13. (Currently Amended) Method as in claim 12, ~~characterized in that~~ wherein said instrumentation of the program code is based on explicit handling of a variable or a register representing the trace print and/or on the call to specialized routines and/or on the use of specialized instructions of the execution machine.

14. (Currently Amended) Method as in claim 12, ~~characterized in that~~ wherein said complementary program data is coded in tables which associate program points with a code defining ~~the~~ an operation to be performed, and which are only consulted by the execution machine when executing particular instructions.

15. (Currently Amended) Method as in claim 14, ~~characterized in that~~ wherein said particular instructions are branches and/or writing in non-volatile memory and/or calls to certain program routines and/or certain cryptographic operations.

16. (Currently Amended) Method as in claim 1, ~~characterized in that~~ wherein the expected trace print values and trace print adjustment values at given program points are determined by static analysis of the program ~~code (source or object)~~ which can simulate ~~the~~ an unwinding of some loops and recursions and which can modify the program ~~code~~ to make the trace print values predictable and/or to check these values.

17. (Currently Amended) Method as in ~~any of claims 4 to 16~~ claim 9 ~~characterized in that~~ wherein for the purpose of said analysis, information is provided concerning trace

print updating (program points and type of execution observations at this program point) and/or trace print adjustment (program points where the trace print must be adjusted to a certain value) and/or trace print assignment (program points where the trace print must be forced to a value) and/or trace print controlling (program points where the trace print must be checked), this information:

- being determined automatically ~~according to the method as in any of claims 6 to 11, and/or~~
- being given in the form of directives consisting of instructions placed in the program code and operating on the trace print (such as program routine calls, whether or not taking any integer as argument) and/or being given in the form of tables complementary to the program,
- and able to be completed and/or modified in accordance with the values calculated by said analysis.

18. (Currently Amended) Method as in claim 17, ~~characterized in that~~ <u>wherein</u> for each program routine, the expected trace print values are determined by the following operating sequence:

- Initialising all the program points to be explored with the singleton formed of the first program routine instruction.

- Memorizing, at the program routine entry point, a trace print value equal to the initial trace print value given.
- For as long as said set of program points to be explored is not void:
  - Extracting a program point (point of origin) from said set of program points to be explored,
  - For each of the resulting possible program points after execution of the instruction (target points):
    * If the target point contains a trace print assignment and if this target point has not yet been explored, memorizing at the target point the trace print value defined by the assignment.
    * If the target point does not contain a trace print assignment and if this target point has already been explored, inserting between the instruction at the point of origin and the instruction at the target point a trace print adjustment which sends the trace print value at the point of origin onto the trace print value memorized at the target point.
    * If the target point does not contain a trace print assignment and if this target point has not yet been explored, memorizing at the target point the trace print value at the point of origin,

optionally modified by a trace print update if one exists between the point of origin and the target point.

*If the target point has not yet been explored, adding said target point in said set of program points to be explored.

19. (Currently Amended) Method as ~~in any of claims 12 to 18~~ claimed in claim 17, ~~characterized in that~~ wherein ~~firstly~~ the trace print concerns complete execution of the program (including with program routine calls) from its entry points, ~~and secondly the~~ said ~~method as in claim 17 is~~ being applied to a set of routines by treating the instructions of static program routine call as unconditional branches on the first instruction of the called program routine, the instructions of dynamic program routine call as conditional branches on the first instruction of the corresponding called program routine, and the instructions of return call as branches towards the instructions following immediately after the corresponding call.

20. (Currently Amended) Method as claimed in claim 12, ~~characterized in that~~ wherein the program and/or the execution machine are instrumented so that the trace print is

saved on certain calls to routines (such as those which are not part of the program or cannot be analysed) and is restored on return call.

21. (Currently Amended) Method as claimed in claim 12, ~~characterized in that~~ wherein the program and/or the execution machine are instrumented so that the trace print is adjusted on call and return from certain routines (including routines determined dynamically at the time of call) so that it is equal to:

- on entry of the called program routine: a value which depends on the name and/or signature of the called program routine (such as a value obtained by cryptographic tracing print of the name and/or signature);
- after return in the calling program routine: a value which similarly depends on the name and/or signature of the called program routine, each exception handler concerned by the program routine call (i.e. possibly being affected when an exception is lifted in the called program routine) having to assign the trace print to a determined value.

22. (Currently Amended) Method as claimed in ~~either of claims 3 and 12~~ claim 3, ~~characterized in that~~ wherein if the trace print is updated implicitly by an execution machine:

- trace print setting may be temporarily suspended to avoid unnecessary calculations when executing non-critical code fragments of the program and/or when program status is not considered critical and/or during the execution of certain routines not performing a trace print check;
- trace print setting, if it is not suspended, relates to each executed instruction,
    * including some of its immediate arguments and/or some of ~~the~~ program invariants for this instruction (such as the height of the operand stack or the presence of certain types of values in the operand stack) and/or ~~the~~ choices of branch made if the instruction is a branch,
    * but provided that the executed instruction belongs to a given class of instructions to be observed, said class being fixed for the execution machine or else given by a table associating a Boolean with every instruction code indicating whether the instruction is to be observed, and said table

being specific to different routines and/or different programs.

23. (Currently Amended) Method as in claim 12, ~~characterized in that~~ wherein:

- some operations on the trace print (such as trace print assignment and controlling) are inserted explicitly in the program code;
- some operations on the trace print (such as trace print adjustment) are performed explicitly by the execution machine in relation to complementary program information,
- some operations on the trace print (such as trace print updating) are performed implicitly by the execution machine.

24. (Currently Amended) Method as in claim 12, ~~characterized in that~~ wherein:

- if trace print set and/or check operations are made by program routine calls, the program is accompanied by a library which implements these routines, said library possibly being substituted by a special implementation when loading on an execution platform;
- if the trace print set and check operations are expressed by complementary program information and if the execution platform does not know and/or cannot and/or does not want to

use this information, said information is ignored to enable execution without integrity controlling.

25. (Currently Amended) Method as ~~in either of claims 12 and~~ claim 20, ~~characterized in that~~ wherein the execution machine of the program has specialized instructions for trace print calculation and/or trace print update and/or trace print adjustment and/or trace print assignment and/or trace print controlling and/or trace print saving on calls to routines and trace print restoration on return from a program routine, these instructions appearing explicitly in the program code and/or being used to implement the execution machine.

26. (Currently Amended) Execution system enabling controlling of execution integrity ~~characterized in that~~ wherein said system includes a microprocessor which has specialized instructions for trace print calculation and/or trace print update and/or trace print adjustment and/or trace print assignment and/or trace print controlling and/or trace print saving on calls to routines and trace print restoration on return from a program routine, ~~in accordance with the method as in any of claims 1 to 25~~ wherein said controlling comprises the following steps:

- updating a trace print representing an execution pathway and/or handled data on program execution,
- comparing said trace print (current value, calculated dynamically) with an expected value (fixed statically, equal to a value the trace print should have if program execution is not disturbed) at determined points of the program,
- performing special treatment if the current trace print differs from the expected value.